Our Ref. No. 042390.P8628
Express Mail No.: EL466333353US

UNITED STATES PATENT APPLICATION

FOR

MANAGING A SECURE ENVIRONMENT USING A CHIPSET IN ISOLATED EXECUTION MODE

INVENTORS:

Carl M. Ellison
Roger A. Golliver
Howard C. Herbert
Derrick C. Lin
Francis X. McKeen
Gil Neiger
Ken Reneris
James A. Sutton
Shreekant S. Thakkar
Millind Mittal

PREPARED BY:

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP
12400 Wilshire Blvd., 7th Floor
Los Angeles, CA 90025-1026
(714) 557-3800

BACKGROUND

1. Field of the Invention

This invention relates to microprocessors. In particular, the invention relates to processor security.

2. Description of Related Art

Advances in microprocessor and communication technologies have opened up many opportunities for applications that go beyond the traditional ways of doing business. Electronic commerce (E-commerce) and business-to-business (B2B) transactions are now becoming popular, reaching the global markets at a fast rate. Unfortunately, while modern microprocessor systems provide users convenient and efficient methods of doing business, communicating and transacting, they are also vulnerable to unscrupulous attacks. Examples of these attacks include virus, intrusion, security breach, and tampering, to name a few. Computer security, therefore, is becoming more and more important to protect the integrity of the computer systems and increase the trust of users.

Threats caused by unscrupulous attacks may be in a number of forms. Attacks may be remote without requiring physical accesses. An invasive remote-launched attack by hackers may disrupt the normal operation of a system connected to thousands or even millions of users. A virus program may corrupt code and/or data of a single-user platform.

Existing techniques to protect against attacks have a number of drawbacks. Anti-virus programs can only scan and detect known viruses. Most anti-virus programs use a weak policy in which a file or program is assumed good until proved bad. For many

security applications, this weak policy may not be appropriate. In addition, most anti-virus programs are used locally where they are resident in the platform. This may not be suitable in a group work environment. Security co-processors or smart cards using cryptographic or other security techniques have limitations in speed performance, memory capacity, and flexibility. Redesigning operating systems creates software compatibility issues and causes tremendous investment in development efforts.

BRIEF DESCRIPTION OF THE DRAWINGS

The features and advantages of the present invention will become apparent from the following detailed description of the present invention in which:

Figure 1A is a diagram illustrating a logical operating architecture according to one embodiment of the invention.

Figure 1B is a diagram illustrating accessibility of various elements in the operating system and the processor according to one embodiment of the invention.

Figure 1C is a diagram illustrating a computer system in which one embodiment of the invention can be practiced.

Figure 2 is a diagram illustrating a chipset environment according to one embodiment of the invention.

Figure 3 is a diagram illustrating a chipset circuit shown in Figure 2 according to one embodiment of the invention.

Figure 4 is a flowchart illustrating a process to manage a chipset according to one embodiment of the invention.

Figure 5 is a flowchart illustrating a process to enroll a thread according to one embodiment of the invention.

Figure 6 is a flowchart illustrating a process to withdraw a thread according to one embodiment of the invention.

Figure 7 is a flowchart illustrating a process to write chipset mode according to one embodiment of the invention.

DETAILED DESCRIPTION

In the following description, for purposes of explanation, numerous details are set forth in order to provide a thorough understanding of the present invention. However, it will be apparent to one skilled in the art that these specific details are not required in order to practice the present invention. In other instances, well-known electrical structures and circuits are shown in block diagram form in order not to obscure the present invention.

ARCHITECTURE OVERVIEW

One principle for providing security in a computer system or platform is the concept of an isolated execution architecture. The isolated execution architecture includes logical and physical definitions of hardware and software components that interact directly or indirectly with an operating system of the computer system or platform. An operating system and the processor may have several levels of hierarchy, referred to as rings, corresponding to various operational modes. A ring is a logical division of hardware and software components that are designed to perform dedicated tasks within the operating system. The division is typically based on the degree or level of privilege, namely, the ability to make changes to the platform. For example, a ring-0 is the innermost ring, being at the highest level of the hierarchy. Ring-0 encompasses the most critical, privileged components. In addition, modules in Ring-0 can also access to lesser privileged data, but not vice versa. Ring-3 is the outermost ring, being at the lowest level of the hierarchy. Ring-3 typically encompasses users or applications level and has the least privilege. Ring-1 and ring-2 represent the intermediate rings with decreasing levels of privilege.

Figure 1A is a diagram illustrating a logical operating architecture 50 according to one embodiment of the invention. The logical operating architecture 50 is an abstraction

of the components of an operating system and the processor. The logical operating architecture 50 includes ring-0 10, ring-1 20, ring-2 30, ring-3 40, and a processor nub loader 52. The processor nub loader 52 is an instance of an processor executive (PE) handler. The PE handler is used to handle and/or manage a processor executive (PE) as will be discussed later. The logical operating architecture 50 has two modes of operation: normal execution mode and isolated execution mode. Each ring in the logical operating architecture 50 can operate in both modes. The processor nub loader 52 operates only in the isolated execution mode.

Ring-0 10 includes two portions: a normal execution Ring-0 11 and an isolated execution Ring-0 15. The normal execution Ring-0 11 includes software modules that are critical for the operating system, usually referred to as kernel. These software modules include primary operating system (e.g., kernel) 12, software drivers 13, and hardware drivers 14. The isolated execution Ring-0 15 includes an operating system (OS) nub 16 and a processor nub 18. The OS nub 16 and the processor nub 18 are instances of an OS executive (OSE) and processor executive (PE), respectively. The OSE and the PE are part of executive entities that operate in a secure environment associated with the isolated area 70 and the isolated execution mode. The processor nub loader 52 is a protected bootstrap loader code held within a chipset in the system and is responsible for loading the processor nub 18 from the processor or chipset into an isolated area as will be explained later.

Similarly, ring-1 20, ring-2 30, and ring-3 40 include normal execution ring-1 21, ring-2 31, ring-3 41, and isolated execution ring-1 25, ring-2 35, and ring-3 45, respectively. In particular, normal execution ring-3 includes N applications 42_1 to 42_N and isolated execution ring-3 includes K applets 46_1 to 46_K.

One concept of the isolated execution architecture is the creation of an isolated region in the system memory, referred to as an isolated area, which is protected by both the processor and chipset in the computer system. The isolated region may also be in cache memory, protected by a translation look aside (TLB) access check. Access to this isolated region is permitted only from a front side bus (FSB) of the processor, using special bus (e.g., memory read and write) cycles, referred to as isolated read and write cycles. The special bus cycles are also used for snooping. The isolated read and write cycles are issued by the processor executing in an isolated execution mode. The isolated execution mode is initialized using a privileged instruction in the processor, combined with the processor nub loader 52. The processor nub loader 52 verifies and loads a ring-0 nub software module (e.g., processor nub 18) into the isolated area. The processor nub 18 provides hardware-related services for the isolated execution.

One task of the processor nub 18 is to verify and load the ring-0 OS nub 16 into the isolated area, and to generate the root of a key hierarchy unique to a combination of the platform, the processor nub 18, and the operating system nub 16. The operating system nub 16 provides links to services in the primary OS 12 (e.g., the unprotected segments of the operating system), provides page management within the isolated area, and has the responsibility for loading ring-3 application modules 45, including applets 46_1 to 46_K, into protected pages allocated in the isolated area. The operating system nub 16 may also load ring-0 supporting modules.

The operating system nub 16 may choose to support paging of data between the isolated area and ordinary (e.g., non-isolated) memory. If so, then the operating system nub 16 is also responsible for encrypting and hashing the isolated area pages before evicting the page to the ordinary memory, and for checking the page contents upon restoration of the page. The isolated mode applets 46_1 to 46_K and their data are tamper-resistant and monitor-resistant from all software attacks from other applets, as well as

from non-isolated-space applications (e.g., 42_1 to 42_N), dynamic link libraries (DLLs), drivers and even the primary operating system 12. Only the processor nub 18 or the operating system nub 16 can interfere with or monitor the applet's execution.

Figure 1B is a diagram illustrating accessibility of various elements in the operating system 10 and the processor according to one embodiment of the invention. For illustration purposes, only elements of ring-0 10 and ring-3 40 are shown. The various elements in the logical operating architecture 50 access an accessible physical memory 60 according to their ring hierarchy and the execution mode.

The accessible physical memory 60 includes an isolated area 70 and a non-isolated area 80. The isolated area 70 includes applet pages 72 and nub pages 74. The non-isolated area 80 includes application pages 82 and operating system pages 84. The isolated area 70 is accessible only to elements of the operating system and processor operating in isolated execution mode. The non-isolated area 80 is accessible to all elements of the ring-0 operating system and to the processor.

The normal execution ring-0 11 including the primary OS 12, the software drivers 13, and the hardware drivers 14, can access both the OS pages 84 and the application pages 82. The normal execution ring-3, including applications 42_1 to 42_N, can access only to the application pages 82. Both the normal execution ring-0 11 and ring-3 41, however, cannot access the isolated area 70.

The isolated execution ring-0 15, including the OS nub 16 and the processor nub 18, can access to both of the isolated area 70, including the applet pages 72 and the nub pages 74, and the non-isolated area 80, including the application pages 82 and the OS pages 84. The isolated execution ring-3 45, including applets 46_1 to 46_K, can access only to the application pages 82 and the applet pages 72. The applets 46_1 to 46_K reside in the isolated area 70.

Figure 1C is a diagram illustrating a computer system 100 in which one embodiment of the invention can be practiced. The computer system 100 includes a processor 110, a host bus 120, a memory controller hub (MCH) 130, a system memory 140, an input/output controller hub (ICH) 150, a non-volatile memory, or system flash, 160, a mass storage device 170, input/output devices 175, a token bus 180, a motherboard (MB) token 182, a reader 184, and a token 186. The MCH 130 may be integrated into a chipset that integrates multiple functionalities such as the isolated execution mode, host-to-peripheral bus interface, memory control. Similarly, the ICH 150 may also be integrated into a chipset together or separate from the MCH 130 to perform I/O functions. For clarity, not all the peripheral buses are shown. It is contemplated that the system 100 may also include peripheral buses such as Peripheral Component Interconnect (PCI), accelerated graphics port (AGP), Industry Standard Architecture (ISA) bus, and Universal Serial Bus (USB), etc.

The processor 110 represents a central processing unit of any type of architecture, such as complex instruction set computers (CISC), reduced instruction set computers (RISC), very long instruction word (VLIW), or hybrid architecture. In one embodiment, the processor 110 is compatible with an Intel Architecture (IA) processor, such as the Pentium™ series, the IA-32™ and the IA-64™. The processor 110 includes a normal execution mode 112 and an isolated execution circuit 115. The normal execution mode 112 is the mode in which the processor 110 operates in a non-secure environment, or a normal environment without the security features provided by the isolated execution mode. The isolated execution circuit 115 provides a mechanism to allow the processor 110 to operate in an isolated execution mode. The isolated execution circuit 115 provides hardware and software support for the isolated execution mode. This support includes configuration for isolated execution, definition of an isolated area, definition (e.g.,

decoding and execution) of isolated instructions, generation of isolated access bus cycles, and generation of isolated mode interrupts.

In one embodiment, the computer system 100 can be a single processor system, such as a desktop computer, which has only one main central processing unit, e.g. processor 110. In other embodiments, the computer system 100 can include multiple processors, e.g. processors 110, 110a, 110b, etc., as shown in Figure 1C. Thus, the computer system 100 can be a multi-processor computer system having any number of processors. For example, the multi-processor computer system 100 can operate as part of a server or workstation environment. The basic description and operation of processor 110 will be discussed in detail below. It will be appreciated by those skilled in the art that the basic description and operation of processor 110 applies to the other processors 110a and 110b, shown in Figure 1C, as well as any number of other processors that may be utilized in the multi-processor computer system 100 according to one embodiment of the present invention.

The processor 110 may also have multiple logical processors. A logical processor, sometimes referred to as a thread, is a functional unit within a physical processor having an architectural state and physical resources allocated according to some partitioning policy. Within the context of the present invention, the terms "thread" and "logical processor" are used to mean the same thing. A multi-threaded processor is a processor having multiple threads or multiple logical processors. A multi-processor system (e.g., the system comprising the processors 110, 110a, and 110b) may have multiple multi-threaded processors.

The host bus 120 provides interface signals to allow the processor 110 or processors 110, 100a, and 110b to communicate with other processors or devices, e.g., the MCH 130. In addition to normal mode, the host bus 120 provides an isolated access

bus mode with corresponding interface signals for memory read and write cycles when the processor 110 is configured in the isolated execution mode. The isolated access bus mode is asserted on memory accesses initiated while the processor 110 is in the isolated execution mode. The isolated access bus mode is also asserted on instruction pre-fetch and cache write-back cycles if the address is within the isolated area address range and the processor 110 is initialized in the isolated execution mode. The processor 110 responds to snoop cycles to a cached address within the isolated area address range if the isolated access bus cycle is asserted and the processor 110 is initialized into the isolated execution mode.

The MCH 130 provides control and configuration of memory and input/output devices such as the system memory 140 and the ICH 150. The MCH 130 provides interface circuits to recognize and service isolated access assertions on memory reference bus cycles, including isolated memory read and write cycles. In addition, the MCH 130 has memory range registers (e.g., base and length registers) to represent the isolated area in the system memory 140. Once configured, the MCH 130 aborts any access to the isolated area that does not have the isolated access bus mode asserted.

The system memory 140 stores system code and data. The system memory 140 is typically implemented with dynamic random access memory (DRAM) or static random access memory (SRAM). The system memory 140 includes the accessible physical memory 60 (shown in Figure 1B). The accessible physical memory includes a loaded operating system 142, the isolated area 70 (shown in Figure 1B), and an isolated control and status space 148. The loaded operating system 142 is the portion of the operating system that is loaded into the system memory 140. The loaded OS 142 is typically loaded from a mass storage device via some boot code in a boot storage such as a boot read only memory (ROM). The isolated area 70, as shown in Figure 1B, is the memory area that is defined by the processor 110 when operating in the isolated execution mode.

Access to the isolated area 70 is restricted and is enforced by the processor 110 and/or the MCH 130 or other chipset that integrates the isolated area functionalities. The isolated control and status space 148 is an input/output (I/O)-like, independent address space defined by the processor 110 and/or the MCH 130. The isolated control and status space 148 contains mainly the isolated execution control and status registers. The isolated control and status space 148 does not overlap any existing address space and is accessed using the isolated bus cycles. The system memory 140 may also include other programs or data which are not shown.

The ICH 150 represents a known single point in the system having the isolated execution functionality. For clarity, only one ICH 150 is shown. The system 100 may have many ICH's similar to the ICH 150. When there are multiple ICH's, a designated ICH is selected to control the isolated area configuration and status. In one embodiment, this selection is performed by an external strapping pin. As is known by one skilled in the art, other methods of selecting can be used, including using programmable configuring registers. The ICH 150 has a number of functionalities that are designed to support the isolated execution mode in addition to the traditional I/O functions. In particular, the ICH 150 includes an isolated bus cycle interface 152, the processor nub loader 52 (shown in Figure 1A), a digest memory 154, a cryptographic key storage 155, an isolated execution logical processor manager 156, and a token bus interface 159.

The isolated bus cycle interface 152 includes circuitry to interface to the isolated bus cycle signals to recognize and service isolated bus cycles, such as the isolated read and write bus cycles. The processor nub loader 52, as shown in Figure 1A, includes a processor nub loader code and its digest (e.g., hash) value. The processor nub loader 52 is invoked by execution of an appropriate isolated instruction (e.g., Iso_Init) and is transferred to the isolated area 70. From the isolated area 80, the processor nub loader 52 copies the processor nub 18 from the system flash memory (e.g., the processor nub code

18 in non-volatile memory 160) into the isolated area 70, verifies and logs its integrity, and manages a symmetric key used to protect the processor nub's secrets. In one embodiment, the processor nub loader 52 is implemented in read only memory (ROM). For security purposes, the processor nub loader 52 is unchanging, tamper-resistant and non-substitutable. The digest memory 154, typically implemented in RAM, stores the digest (e.g., hash) values of the loaded processor nub 18, the operating system nub 16, and any other critical modules (e.g., ring-0 modules) loaded into the isolated execution space. The cryptographic key storage 155 holds a symmetric encryption/decryption key that is unique for the platform of the system 100. In one embodiment, the cryptographic key storage 155 includes internal fuses that are programmed at manufacturing. Alternatively, the cryptographic key storage 155 may also be created with a random number generator and a strap of a pin. The isolated execution logical processor manager 156 manages the operation of logical processors operating in isolated execution mode. In one embodiment, the isolated execution logical processor manager 156 includes a logical processor count register that tracks the number of logical processors participating in the isolated execution mode. The token bus interface 159 interfaces to the token bus 180. A combination of the processor nub loader digest, the processor nub digest, the operating system nub digest, and optionally additional digests, represents the overall isolated execution digest, referred to as isolated digest. The isolated digest is a fingerprint identifying the ring-0 code controlling the isolated execution configuration and operation. The isolated digest is used to attest or prove the state of the current isolated execution.

The non-volatile memory 160 stores non-volatile information. Typically, the non-volatile memory 160 is implemented in flash memory. The non-volatile memory 160 includes the processor nub 18. The processor nub 18 provides the initial set-up and low-level management of the isolated area 70 (in the system memory 140), including verification, loading, and logging of the operating system nub 16, and the management of

the symmetric key used to protect the operating system nub's secrets. The processor nub 18 may also provide application programming interface (API) abstractions to low-level security services provided by other hardware. The processor nub 18 may also be distributed by the original equipment manufacturer (OEM) or operating system vendor (OSV) via a boot disk.

The mass storage device 170 stores archive information such as code (e.g., processor nub 18), programs, files, data, applications (e.g., applications 42_1 to 42_N), applets (e.g., applets 46_1 to 46_K) and operating systems. The mass storage device 170 may include compact disk (CD) ROM 172, floppy diskettes 174, and hard drive 176, and any other magnetic or optical storage devices. The mass storage device 170 provides a mechanism to read machine-readable media. When implemented in software, the elements of the present invention are the code segments to perform the necessary tasks. The program or code segments can be stored in a processor readable medium or transmitted by a computer data signal embodied in a carrier wave, or a signal modulated by a carrier, over a transmission medium. The "processor readable medium" may include any medium that can store or transfer information. Examples of the processor readable medium include an electronic circuit, a semiconductor memory device, a ROM, a flash memory, an erasable programmable ROM (EPROM), a floppy diskette, a compact disk CD-ROM, an optical disk, a hard disk, a fiber optical medium, a radio frequency (RF) link, etc. The computer data signal may include any signal that can propagate over a transmission medium such as electronic network channels, optical fibers, air, electromagnetic, RF links, etc. The code segments may be downloaded via computer networks such as the Internet, an Intranet, etc.

I/O devices 175 may include any I/O devices to perform I/O functions. Examples of I/O devices 175 include a controller for input devices (e.g., keyboard, mouse, trackball,

pointing device), media card (e.g., audio, video, graphics), a network card, and any other peripheral controllers.

The token bus 180 provides an interface between the ICH 150 and various tokens in the system. A token is a device that performs dedicated input/output functions with security functionalities. A token has characteristics similar to a smart card, including at least one reserved-purpose public/private key pair and the ability to sign data with the private key. Examples of tokens connected to the token bus 180 include a motherboard token 182, a token reader 184, and other portable tokens 186 (e.g., smart card). The token bus interface 159 in the ICH 150 connects through the token bus 180 to the ICH 150 and ensures that when commanded to prove the state of the isolated execution, the corresponding token (e.g., the motherboard token 182, the token 186) signs only valid isolated digest information. For purposes of security, the token should be connected to the digest memory.

A CHIPSET CIRCUIT TO MANAGE A SECURE PLATFORM

The overall architecture discussed above provides a basic insight into a hierarchical executive architecture to manage a secure platform. The elements shown in Figures 1A, 1B, and 1C are instances of an abstract model of this hierarchical executive architecture. The implementation of this hierarchical executive architecture is a combination of hardware and software. In what follows, the processor executive, the processor executive handler, and the operating system executive are abstract models of the processor nub 18, the processor nub loader 52, and the operating system nub 16 (Figures 1A, 1B, and 1C), respectively.

Figure 2 is a diagram illustrating a chipset environment 200 according to one embodiment of the invention. The chipset environment 200 includes a chipset 205 and executive entities 220.

The chipset 205 is an integrated device that provides the necessary infrastructure for the isolated execution mode in the platform. In one embodiment, the chipset 205 is the input/output controller hub (ICH) 150 shown in Figure 1C. The chipset 205 includes a chipset circuit 210. The chipset circuit 210 includes the functionalities to allow a platform to operate in a secure environment. The secure environment is associated with the isolated memory area (e.g., the isolated memory area 70 shown in Figure 1C) which is accessible to a processor in the platform. The platform may have one or multiple processors (e.g., the processor 110 in Figure 1C). Each processor may operate in one of a normal execution mode and an isolated execution mode. Each processor may have one or multiple logical threads. The chipset 205 keeps tracks of the number of logical threads operating in the execution mode for the entire platform, whether the platform is a uni- or multi-processor system.

The executive entities 220 include a number of executives that are designed to operate in the secure environment. The executive entities 220 are associated with the isolated memory area 70 (Figure 1C). The executive entities 220 include a processor executive (PE) handler 230, a PE 240, and an operating system executive (OSE) 250. The PE handler 230 handles the PE 240. The PE 240 handles the OSE 250. The OSE 250 interfaces to a subset of the operating system (OS) 260 running on the platform. As mentioned above, the PE handler 230, the PE 240, and the OSE 250 are abstract models of the processor nub loader 52, the processor nub 18, and the operating system nub 16, respectively, shown in Figures 1A-1C.

Figure 3 is a diagram illustrating the chipset circuit 210 shown in Figure 2 according to one embodiment of the invention. The chipset circuit 210 includes an initialization storage 310, a PE handler storage 320, a thread count storage 330, a thread count updater 340, a mode storage 350, a mode write circuit 360, an identifier log storage 370, a log lock storage 375, a lock circuit 378, a fused key storage 380, a scratch storage

390, and a status storage 395. The initialization storage 310, the PE handler storage 320, the thread count storage 330, the mode storage 350, the identifier log storage 370, the log lock storage 375, the fused key storage 380, the scratch storage 390, and the status storage 390 are all mapped into an input/output address space accessible to the processor 110 in isolated execution mode. The thread count updater 340, the mode write circuit 360, and the lock circuit 378 are invoked or triggered when the corresponding storage is accessed.

The initialization storage 310 is used to initialize the chipset 205 for isolated execution mode. The initialization storage is accessed by an initialization storage access signal 305. The initialization storage access signal 305 represents an input/output bus access cycle decoded to the initialization storage address as generated by one of the processors in the system (e.g., processor 110 in Figure 1C).

The PE handler storage 320 stores PE handler data 322. The PE handler data 322 include a PE handler image 323, a PE handler identifier 325, a PE handler size 327, and a PE handler address 329. The PE handler image 323 is to be loaded into the isolated memory area 70 after the chipset 205 is initialized. The loaded PE handler image 323 corresponds to the PE handler 230 in the executive entities 220 (Figure 2).

The thread count storage 330 stores a thread count 332 to indicate the number of threads currently operating in the isolated execution mode. The thread count 332 is useful for a thread to know if it is the first thread to initialize the platform in the isolated execution mode. The thread count 332 keeps track of the number of threads so that the executive entities 220 can determine the available resources. This is especially useful for scheduling and load balancing tasks performed by the OSE 250 and the OS 260. In addition, the thread count 332 helps in the orderly initialization and participation of the threads in the isolated execution mode. The thread count 332 is updated when the

initialization storage 310 is accessed and when there is no failure. The limit comparator 335 compares the thread count 332 with high and low limits to determine if the thread count has exceeded the maximum or minimum available threads. When a maximum or high limit is exceeded, a failure or fault condition is generated. When a minimum or low limit has been reached, the chipset 205 is initialized to the initial conditions. The updated thread count is one of an incremented thread count and a decremented thread count. A current thread count is returned when there is a failure as indicated by a failure mode in the mode storage 350 as will be described later. The incremented thread count is returned when one of the threads enrolls in the isolated execution mode. The decremented thread count is returned when one of the enrolled threads withdraws from the isolated execution mode.

The mode storage 350 stores a chipset mode 352. The mode circuit 360 writes the chipset mode 352 into the mode storage 350. The chipset mode 352 indicates a mode of operation of the chipset 205. This mode of operation includes an initialization waiting mode, a PE initialization in-progress mode, a PE initialization completion mode, an OSE loaded mode, a closing mode, and a failure mode. The initialization waiting mode indicates that the chipset is waiting for initialization. The PE initialization in-progress mode indicates that the PE is being executed. The PE initialization completion mode indicates that the PE is completed. The OSE loaded mode indicates the OSE has been loaded. The closing mode indicates that the isolated execution mode is closed. The failure mode indicates that there is a failure.

The identifier log storage 370 stores identifiers 372_1 to 372_N of the executive entities operating in the isolated execution mode. The identifiers 372_1 to 372_N are read-only when in lock. The log lock storage 375 stores a lock pattern 376 to indicate which of the identifiers 372_1 to 372_N is in lock. To be "in lock" means that the corresponding

identifier cannot be modified or written. The identifiers 372_1 to 372_N are locked based on the lock pattern 376.

The fused key storage 380 stores a fused key 385. The fused key 385 is used in handling the executive entities 220. The fused key 385 is returned when the fused key storage 380 is read in an initialization waiting mode as set by the mode storage 350 which will be described later. The fused key 385 is programmed at manufacturing time to a random value 387.

The scratch storage 390 stores isolated settings 392 used to configure the isolated execution mode. The isolated settings 392 include an isolated base value, an isolated mask value, and a processor executive entry address. The isolated base and mask values define the isolated memory area 70, and are essentially the same as the isolated base and mask values as stored in the processor 110 and other chipsets (e.g., the MCH 130 in Figure 1C).

The status storage 395 stores a status value 396 of an isolated unlock pin 397 used in restoring a root key from the fused key 385.

Figure 4 is a flowchart illustrating a process 400 to manage a chipset according to one embodiment of the invention.

Upon START, the process 400 boots up the platform and configures the platform in the isolated execution mode (Block 410). Then, the process 400 writes the chipset mode as the initialization waiting mode (Block 420). Next, the process 400 loads the PE handler image into the isolated memory area (Block 430). The loaded PE handler image becomes the PE handler. Next, the process 400 invokes the PE handler and other executive entities (Block 440).

Then, the process 400 determines if the thread operation is a thread enrolment or a thread withdrawal. If it is a thread enrolment, the process 400 enrolls the thread (Block 460). If it is a thread withdrawal, the process 400 withdraws the thread (Block 470). Then, the process 400 is terminated.

Figure 5 is a flowchart illustrating the process 460 to enroll a thread shown in Figure 4 according to one embodiment of the invention.

Upon START, the process 460 accesses the enrollment storage (Block 510). Then, the process 460 increments the thread count (Block 520). Then, the process 460 determines if the thread count has exceeded a high limit or has reached a maximum value (Block 530). If yes, the process 460 writes the chipset mode as a failure mode (Block 540) and proceeds to Block 550. Otherwise, the process 460 determines if the failure mode is set or if there is a failure (Block 560). If yes, the process 460 returns the current thread count (Block 550) and is then terminated. Otherwise, the process 460 returns the incremented thread count (Block 570) and is then terminated.

Figure 6 is a flowchart illustrating the process 470 to withdraw a thread shown in Figure 4 according to one embodiment of the invention.

Upon START, the process 470 accesses the withdrawal storage (Block 610). Then, the process 470 decrements the thread count (Block 620). Then, the process 470 determines if the thread count has exceeded a low limit or has reached a minimum value (Block 630). If yes, the process 470 resets the chipset storage to initial values (Block 640) and proceeds to Block 650. Otherwise, the process 470 determines if the failure mode is set or if there is a failure (Block 660). If yes, the process 470 returns the current thread count (Block 650) and is then terminated. Otherwise, the process 470 returns the decremented thread count (Block 670) and is then terminated.

Figure 7 is a flowchart illustrating a process 440 to write the chipset mode shown in Figure 4 according to one embodiment of the invention.

Upon START, the process 440 reads the fused key storage to obtain the fused key (Block 710). Then, the process 440 determines if the initialization waiting mode is set (Block 715). If not, the process 440 writes the chipset mode as a failure mode (Block 770) and is then terminated. Otherwise, the process 440 returns the fused key, initializes, and invokes the processor executive (Block 720). During the execution of the processor executive, if there is any failure, the process 440 proceeds to Block 770.

Next, the process 440 writes the chipset mode as PE initialization in progress (Block 725). Then, the process 440 determines if the PE operation is completed (Block 730). If not, the process 440 returns to Block 730 waiting for the PE to complete its task. Otherwise, the process 440 writes the chipset mode as PE initialization completion (Block 735). Then, the process 440 initializes and invokes the operating system executive (OSE) (Block 740). During the execution of the OSE, if there is any failure, the process 440 proceeds to Block 770.

Next, the process 440 determines if the OSE has been loaded or has completed its task (Block 745). If not, the process 440 returns to Block 745 waiting for the OSE to be loaded or to complete its task. Otherwise, the process 440 writes the chipset mode as OSE loaded (Block 750). Then, the process 440 determines if the isolated execution mode is closed (Block 755). If yes, the process 440 writes the chipset mode as closing (Block 760) and is then terminated. Otherwise, the process 440 is terminated.

While this invention has been described with reference to illustrative embodiments, this description is not intended to be construed in a limiting sense. Various modifications of the illustrative embodiments, as well as other embodiments of the

invention, which are apparent to persons skilled in the art to which the invention pertains are deemed to lie within the spirit and scope of the invention.